Detecção de estresse e esforço físico a partir de sinais fisiológicos

Gabriel A. Streicher
Departamento de Computação (DComp)
Universidade Federal de São Carlos (UFSCar)
18052-780, Sorocaba, São Paulo, Brasil
gabriel.streicher@estudante.ufscar.br

***Resumo*—Este trabalho investiga o uso de métodos de aprendizado supervisionado para detecção de estados fisiológicos a partir de séries temporais coletadas por sensores de dispositivos vestíveis. Serão aplicadas técnicas de processamento de dados, análise exploratória, experimentos com modelos e, por fim, avaliação dos resultados. Este projeto faz parte da avaliação da disciplina de Aprendizado de Máquina da UFSCar em 2025.**

***Palavras-chave*-Aprendizado de Máquina; séries temporais; sensores vestíveis; sinais fisiológicos;**

I. Introdução

O monitoramento de sinais fisiológicos por meio de dispositivos vestíveis se tornou uma prática muito comum nos dias atuais. A partir desses dados, modelos de aprendizado de máquina conseguem analisar e acompanhar a saúde do usuário de forma automática. O objetivo desse trabalho é o desenvolvimento de um modelo capaz de classificar o estado fisiológico de usuários (STRESS, ANAEROBIC e AEROBIC) a partir desses dados coletados por sensores vestíveis.

II. Trabalhos Relacionados

Com a ascenção de smartwatches e outros aparelhos vestíveis, diversos estudos apresentaram discussões de como combinar esses dados com algoritmos de aprendizado de máquina para detectar estados fisiológicos e quais os melhores métodos para se utilizar nessa combinação. Em particular, o artigo de Hongn et al. [1] analisou dados muito semelhantes ao deste trabalho e identificou que o método Gradient Boosting foi o que apresentou melhor resultado nesse contexto.

Diversos outros métodos de aprendizado supervisionado foram testados e avaliados nesse tipo de conjunto de dados, destacando-se também algoritmos como o Random Forest e Redes Neurais [2]. No entanto, o maior foco desse trabalho será o Gradiente Boosting e, adicionalmente, será brevemente avaliado os seguintes métodos: K-Vizinhos Mais Próximos, Regressão Logistica, Redes Neurais, SVM e Naive Bayes.

III. Dados e pré-processamento

O conjunto de dados utilizado nesse trabalho é composto por duas partes principais:

1) Arquivo tabular que contém informações do perfil do usuário;
2) Diretório de séries temporais para cada sensor utilizado.

O arquivo tabular apresenta tanto atributos numéricos, como a idade, altura e peso. Quanto atributos categóricos, por exemplo, genêro e se pratica atividade física regularmente. Para as séries temporais, há um diretório contendo os arquivos csv de cada um dos sensores disponíveis:

- **EDA**: Atividade eletrodérmica
- **TEMP**: Temperatura da pele
- **BVP**: Volume de pulso sanguíneo
- **HR**: Frequência cardíaca
- **IBI**: Intervalo entre batimentos
- **ACC**: Acelerômetro

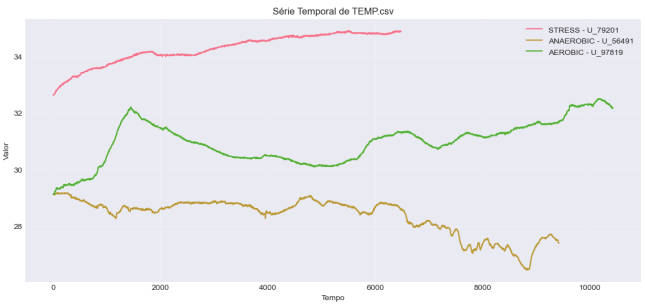

Figure 1. Amostra do sensor TEMP para cada classe

A. Informações dos usuários

Ao analisar esses dados, o primeiro problema encontrado é a ausência de alguns valores. Essa ausência é representada pelo caractere '-' e pode ser encontrado, por exemplo, no atributo "Does physical activity regularly?" (Figura 2).

Nesse caso, o valor foi armazenado como NaN (Not a Number) e posteriormente substituído pela média em valores numéricas e moda em valores categóricos.

B. Sensores

Ao fazer a análise exploratória dos sensores foi possível identificar diversos valores fora do padrão. Já que esses sensores no conjunto de dados eram séries temporais, foi necessário selecionar características que encapsulariam de

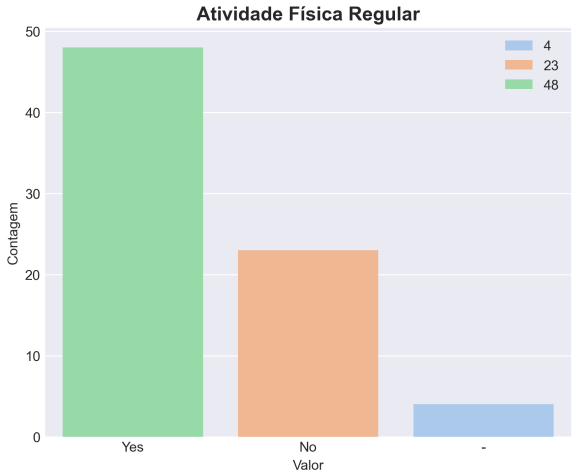

Figure 2. Distribuição do atributo: ”Does physical activity regularly?”

maneira quantitativa os aspectos mais importantes dos dados das séries temporais. As características escolhidas foram: média, mínimo, máximo, mediana e desvio padrão.

Para extrair essas características, foram removidas a primeira e segunda linha dos dados pois essas representavam o timestamp e a frequência, respectivamente. Além disso, caso o valor no arquivo csv não fosse numérico, foi transformado em NaN.

Depois de extraídas as características foi possível analisar cada sensor individualmente. Primeiro, o sensor ACC apresenta 3 colunas em sua série temporal, correspondentes ao eixo x, y e z da aceleração. Em vez de tratar cada eixo separadamente, optei por interpretar essas três dimensões como um vetor de aceleração e calcular sua magnitude. Dessa forma, o valor é unificado em apenas um ponto e retira o efeito da direção nos dados que não importa para o nosso objetivo.

$$Magnitude(t) = \sqrt{X(t)^2 + Y(t)^2 + Z(t)^2}$$

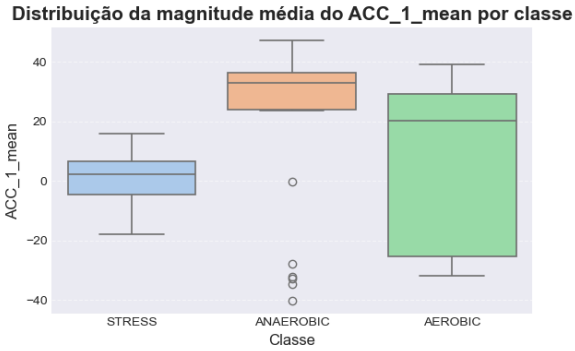

Figure 3. Box plot: Eixo 1 (0-2) do acelerômetro

Além disso, no sensor ACC foi decidido criar uma *feature* especial chamada de energia. Enquanto a magnitude representa a amplitude da aceleração (linear), o seu quadrado é proporcional à potência cinética envolvida no movimento.

$$ACC_{energy} = Magnitude^2$$

Entretanto, antes de transformar e calcular a magnitude, foi possível encontrar um *outlier* no terceiro eixo de um dos usuários. Foi observado que o eixo Z permanecia constante em um valor alto durante longos intervalos, indicando falhas do sensor. Por isso, nesse caso (ou em qualquer outro que apresentar este tipo de outlier) foi decidido ignorar essa coluna e calcular a magnitude apenas com os eixos funcionais.

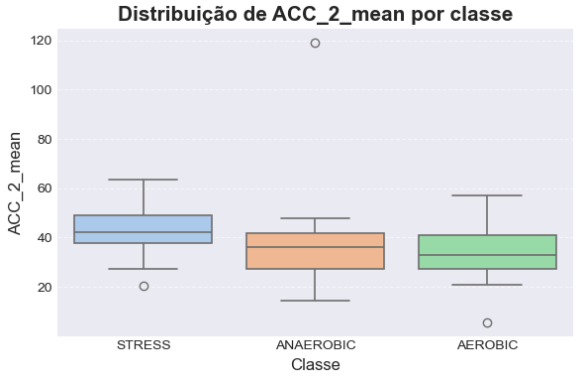

Figure 4. Box plot: Eixo 2 (0-2) do acelerômetro

Para o sensor TEMP, também foi identificado alguns valores absurdos. Nesse caso, esses valores foram substituídos pela média dos valores no conjunto de dados.

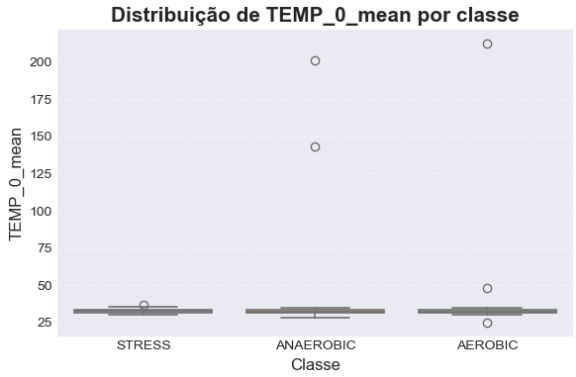

Figure 5. Box plot: Temperatura da pele

Vale destacar que, além das operações de substituir os valores numéricos faltantes utilizando o *SimpleImputer* do *scikit-learn*, também foi utilizado o *LabelEncoder* para transformar os dados categóricos em numéricos.

IV. Protocolo Experimental

Para garantir e avaliar a generalização do modelo, o conjunto de dados rotulados foi dividido em 80% para treino e 20% para validação. Além disso para cada algoritmo foi utilizada validação cruzada estratificada com k=3 durante o ajuste de hiperparâmetros e a métrica de *scoring* 'roc_auc_ovr'.

Todo o processo foi encapsulado dentro de um pipeline do scikit-learn. Sendo possível ativar ou desativar a normalização dos atributos e a redução de dimensionalidade:

$$Scaler(opcional) \rightarrow PCA(opcional) \rightarrow Metodo$$

Após encontrado os melhores hiperparâmetros com a busca em grade, os modelos foram treinados e foi gerada uma curva de aprendizado variando o tamanho do conjunto de treino. A partir desse gráfico é possível observar *overfitting* ou *underfitting*. A curva do método SVM pode ser visualizada na Figura 5.

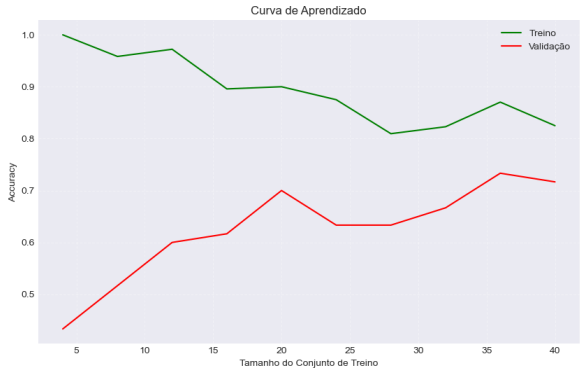

Figure 6. Curva de aprendizado do método SVM.

Outras curvas podem ser visualizadas na implementação do projeto. Neste relatório o foco será o algoritmo XGBoost. Por este modelo ser bastante complexo e a quantidade de dados ser pequena, foi decidido usar hipeparâmetros bastante conservadores na busca em grade visando evitar o overfitting. Para isso foram utilizadas árvores bastante rasas e adicionadas dois tipos de regularização.

Table I
Espaço de busca e hiperparâmetros selecionados no Grid Search.

Hiperparâmetro	Espaço de Busca	Melhor Valor
`n_estimators`	{20, 25, 50, 100}	**20**
`max_depth`	{2, 3, 4}	**2**
`learning_rate`	{0.02, 0.05}	**0.02**
`subsample`	{0.8}	**0.8**
`colsample_bytree`	{0.8}	**0.8**
`L2 Regularization` (λ)	{1, 2, 3}	**1**
`L1 Regularization` (α)	{0.5, 1}	**0.5**

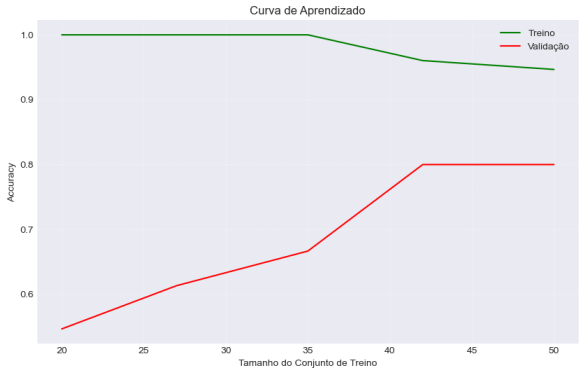

Figure 7. Curva de aprendizado do método XGBoost.

V. Resultados

Depois de treinado o modelo com os hiperparâmetros encontrados, foram obtidas pontuações AUC de 0.985 e 0.979 no conjunto de treinamento e conjunto de validação, respectivamente. Após o envio no Kaggle, foi obtido um score público de 0.99305. Entretanto, anteriormente foi feita uma submissão com hiperparâmetros menos conservadores que, apesar de apresentar uma AUC menor no conjunto de validação (0.96), conseguiu atingir 1.00 no score público do Kaggle.

Table II
Hiperparâmetros score 1.000.

Hiperparâmetro	Valor
`n_estimators`	**600**
`max_depth`	**7**
`learning_rate`	**0.05**
`subsample`	**0.8**
`colsample_bytree`	**0.8**
`L2 Regularization` (λ)	**None**
`L1 Regularization` (α)	**None**

Com o XGBoost também é possível visualizar a importância de cada *feature* para a tomada de decisão do modelo. Na Figura 7 é notado que o atributo mais importante foi o ACC_Energy, o que comprova o sucesso pro pré-processamento dos dados.

Quanto aos outros modelos, na tabela abaixo estão as pontuações AUC obtidas no conjunto de validação de cada um deles. Em alguns foi possível ter um resultado melhor com o uso do PCA e em outros sem o PCA. Ademais, é interessante notar que o algoritmo K-vizinhos mais próximos obteve um *score* maior sem a utilização do PCA. Algo que não era esperado por conta da alta dimensionalidade dos dados de entrada.

VI. Estratégia Final

Por fim, para o envio no Kaggle, também foi feita uma submissão onde o modelo foi treinado com todos os

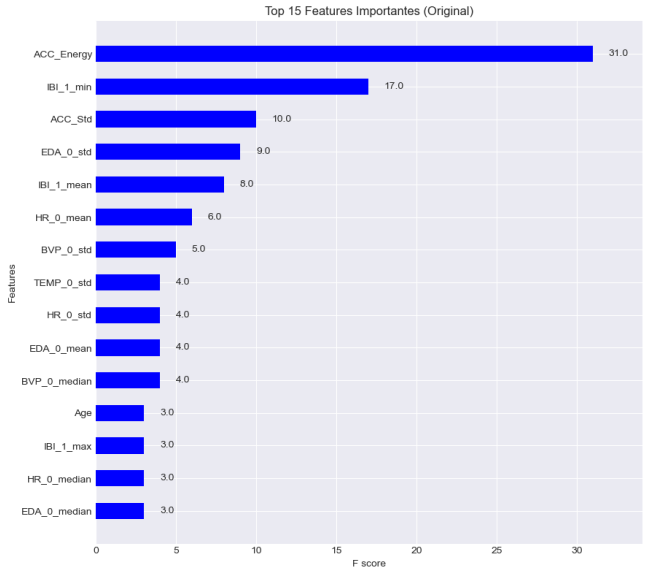

Figure 8. Features mais importantes.

Table III
PONTUAÇÃO AUC

Método	**Score**
KNN	**0.781**
Naive Bayes	**0.617**
Regressão Logística	**0.759**
Rede Neural	**0.745**
SVM	**0.785**

dados disponíveis de treinamento (sem separar a partição de validação). Nesse caso, surpreendentemente obteve um score menor no ranking público do Kaggle: 0.97569. Como mencionado anteriormente, o mesmo modelo treinado com menos dados obteve uma pontuação igual a 0.99305.

Entretanto, apesar da pontuação menor, essa estratégia foi selecionada para ser utilizada no placar privado junto com o modelo gerado pelos valores da Tabela 2. Apesar deste modelo ter uma pontuação maior no placar público, é um modelo mais arriscado e pode ter dado sorte na partição de teste escolhida. Como podemos escolher duas submissões para serem avaliadas no placar privado, decidi optar por uma opção segura e uma de maior risco.

VII. CONCLUSÃO

Este trabalho demonstrou a eficácia do uso de aprendizado de máquina supervisionado para classificar estados fisiológicos. A análise exploratória e o pré-processamento revelaram-se etapas vitais, com destaque para a engenharia de atributos no sensor de acelerômetro. A criação da feature ”Energia” provou ser o fator determinante para a performance do classificador.

Os resultados confirmam que, mesmo com um conjunto de dados limitado, a combinação de modelos baseados em árvores (Gradient Boosting) com um pré-processamento robusto é capaz de apresentar bons resultados na identificação de padrões fisiológicos complexos.

Pensando na extensão do trabalho, seria interessante implementar técnicas de *Data Augmentation* nas séries temporais antes de extrair as features. Isso criaria dados sintéticos para treinar o XGBoost, ajudando a regularizar o modelo e evitar overfitting de forma mais robusta.

REFERENCES

[1] Hongn, A., Bosch, F., Prado, L.E. et al. Wearable Physiological Signals under Acute Stress and Exercise Conditions. Sci Data 12, 520 (2025). https://doi.org/10.1038/s41597-025-04845-9

[2] Vos G., Trinh K., Sarnyai Z. et al. Generalizable machine learning for stress monitoring from wearable devices: A systematic literature review. *International Journal of Medical Informatics*, vol. 173, 2023. https://doi.org/10.1016/j.ijmedinf.2023.105026